EV368629614

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Timeline Source

Inventors:

Sohail Baig Mohammed

Shafiq ur Rahman

Alexandre V. Grigorovitch

Xiqiang Zhi

Adil Sherwani

Geoffrey T. Dunbar

Rebecca C. Weiss

Kirt A. Debique

Patrick N. Nelson

Eric H. Rudolph

ATTORNEY'S DOCKET NO.: **MS1-1849US**

TIMELINE SOURCE

TECHNICAL FIELD

[0001] The present invention generally relates to media, and more particularly relates to a timeline source for rendering a media timeline.

BACKGROUND

[0002] Users of computers, such as desktop PCs, set-top boxes, personal digital assistants (PDAs), and so on, have access to an ever increasing amount of media from an ever increasing variety of sources. For example, a user may interact with a desktop PC that executes a plurality of applications to provide media for output, such as home videos, songs, slideshow presentations, and so on. The user may also utilize a set-top box to receive traditional television programming that is broadcast to the set-top box over a broadcast network. Additionally, the set-top box may be configured as a digital video recorder (DVR) such that the user may store the broadcast content in memory on the set-top box for later playback. Further, the user may interact with a wireless phone that executes a plurality of applications such that the user may read and send email, play video games, view spreadsheets, and so forth.

[0003] Because of the wide variety of media sources and the wide variety of computers that may be utilized to provide and interact with media, traditional applications and computers were often configured to specifically address each particular type of media. For example, applications that were executed on a video-game console to output video-games were typically configured to provide an output of the applications to a television, and were not configured to provide the output that could be utilized by other computers and other devices. Therefore, presentation of media that was provided by the different

media sources, such as computers and/or applications, may involve multiple devices and/or applications which may be both time and device intensive. Additionally, multiple applications that were executed on the same computer may be configured specifically to address the particular type of media provided by each respective application. For instance, a first audio playback application may be configured to output media configured as songs. A second audio playback application, however, may be configured for recording and playback in an audio format that is not compatible with the first audio playback application, such as an audio-dictation format.

[0004] A timeline provides a way for a user to define a presentation of media. For example, a media player can play songs chronologically organized into a timeline, which is commonly referred to as a "playlist". Traditional timelines, however, were limited by the types of media sources and computer configurations that were used to provide and interact with media. When desiring the output of media from different applications, for instance, each type of media would require a different timeline which involved the use of different applications. This use of different applications often resulted in an inefficient use of both hardware and software resources of the computer. Additionally, the different timelines made it difficult to coordinate the outputs from the respective timelines, such as where media was being output from the respective timelines concurrently.

[0005] Further, the execution of large timelines may result in the inefficient use of software and/or hardware resources of the computer. When loading a large playlist of songs, for instance, each song in the playlist was loaded. Therefore, the initial loading of the playlist may consume a significant amount of hardware and/or software resources, thereby resulting in a delay in the loading and playing of songs in the playlist.

[0006] Accordingly, there is a continuing need to provide improved rendering of timelines.

SUMMARY

[0007] A timeline source for rendering a media timeline is described. A media timeline provides a technique for a user to define a presentation based on media. The media timeline may be utilized to express groupings and/or combinations of media and provide compositional metadata utilized by the timeline source to provide a presentation of the media described by the media timeline. The timeline source may be configured in a variety of ways to address a variety of considerations, such as to support dynamic changes to the media timeline. In another example, the timeline source may "pre-roll" sources to minimize delays that arise in rendering content from two consecutive media sources. While content from one source is being rendered, for instance, a subsequent source may be loaded and made ready for immediate use.

[0008] In an implementation, a method includes examining a plurality of nodes within a media timeline, where at least one of the nodes reference respective media. The media timeline is for exposure over an application programming interface (API), such as to an application and/or a rendering engine. The media timeline is divided into one or more presentations. Each presentation describes rendering of the media for a particular interval of time.

[0009] In an additional implementation, a method includes receiving a call from an application over an API for rendering a media timeline. The media timeline includes a plurality of nodes, at least one of which references respective media. The media timeline

defines one or more presentations of the media. The media timeline is rendered to output each of the presentations.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. 1 is an illustration of an environment in an exemplary implementation in which a computer provides access to a plurality of media.

[0011] FIG. 2 is a high level block diagram of a system in an exemplary implementation in which the system, implemented in software, includes an application that interacts with a media foundation to control presentation of a plurality of media.

[0012] FIG. 3 is an illustration of an exemplary implementation in which a media timeline is shown as a tree that includes a plurality of nodes that provide for an output of media for a presentation.

[0013] FIG. 4 is an illustration of an exemplary implementation showing a sequence node and a plurality of leaf nodes that are children of the sequence node.

[0014] FIG. 5 is an illustration of an exemplary implementation in which a sequence node and a plurality of nodes that are children of the sequence node include metadata that specifies timing information for execution of the respective plurality of nodes.

[0015] FIG. 6 is an illustration of an exemplary implementation showing a parallel node and a plurality of leaf nodes that are children of the parallel node.

[0016] FIG. 7 is an illustration of a node in exemplary implementation showing examples of metadata that may be included in the node.

[0017] FIG. 8 is an illustration of a node in an exemplary implementation showing metadata included in the node that specifies a plurality of simple effects.

[0018] FIG. 9 is an illustration of an exemplary implementation showing a parallel node that provides a composite effect to the outputs of two or more child nodes.

[0019] FIG. 10 is an illustration of an exemplary implementation in which a transition effect is specified to supply an effect between an output of media referenced by a previous node to an output of media referenced by a subsequent node.

[0020] FIG. 11 is an illustration of an exemplary implementation showing a node having metadata that includes a plurality of effect metadata.

[0021] FIG. 12 is an illustration a media timeline in an exemplary implementation of dynamic loading in which the nodes of the media timeline are loaded based on metadata that is included in the nodes.

[0022] FIG. 13 is an illustration of a media timeline in an exemplary implementation of dynamic loading in which the nodes of the media timeline are defined and implemented on an as-needed basis by a node source.

[0023] FIG. 14 is an illustration of a media timeline in an exemplary implementation in which events are provided by a node such that changes that occur to the media timeline may be communicated to nodes that may be affected by the changes.

[0024] FIG. 15 is an illustration of an exemplary implementation showing a media timeline that includes a sequence node and three leaf nodes described by a Windows® Media Player Playlist file identified by an ASX file extension.

[0025] FIG. 16 is an illustration of an exemplary implementation showing a media timeline that includes a parallel node having two child sequence nodes that are described by an eXecutable Temporal Language (XTL) file.

[0026] FIG. 17 is an illustration of an exemplary implementation showing an output of first and second media over particular time intervals and including an effect to transition between the first and second media.

[0027] FIG. 18 is an illustration of a media timeline in an exemplary implementation that is suitable to implement the cross fade effect of FIG. 17.

[0028] FIG. 19 is a high level block diagram of a system in an exemplary implementation in which the system, implemented in software, includes a media engine, a media session and a timeline source of a media foundation of FIG. 2 to provide a presentation of media to the application.

[0029] FIG. 20 is an illustration of an exemplary implementation in which a media timeline is shown that includes a sequence node and a plurality of leaf nodes.

[0030] FIG. 21 is an illustration of an exemplary implementation showing an output of media described by the media timeline of FIG. 20 over particular intervals of time and topologies generated by the timeline source to provide the described media rendering.

[0031] FIG. 22 is a flow chart illustrating a procedure in an exemplary implementation in which a media timeline is interpreted by the timeline source for rendering of media as described by the media timeline.

[0032] FIG. 23 is a flow chart illustrating a procedure in an exemplary implementation in which generation of a media timeline and interpretation of the media timeline by a timeline source is described.

[0033] FIG. 24 is a flow chart illustrating a procedure in an exemplary implementation showing rendering of the media timeline of FIG. 22 by the timeline source.

[0034] FIG. 25 is an illustration of an exemplary operating environment.

[0035] The same numbers are used throughout the disclosure and figures to reference like components and features.

DETAILED DESCRIPTION

[0036] **Overview**

A timeline source for rendering a media timeline is described. The media timeline provides a technique for a user to define a presentation based on media, such as already existing media (e.g., stored media such as video, songs, documents, and so on) and/or media that is output in "real-time" from a media source, such as streaming audio and/or video. The media timeline may be utilized to express groupings and/or combinations of media and provide compositional metadata utilized by the timeline source that renders the media timeline to provide a final presentation, based on a reference clock, which includes the media described by the media timeline.

[0037] The timeline source is responsible for managing a media presentation which is described by the media timeline and thereby eases the task of obtaining media from multiple sources at the same time. For example, previously, to obtain media from a plurality of sources at the same time, while keeping the media synchronized, involved substantial coding. Instead of providing separate coding for each application that desires the use of the media, the user can define a media timeline which references media sources that provide media to be played in parallel. The media timeline is then given to a timeline source, which interprets and executes the presentation without user intervention. The media timeline may describe complex presentations with multiple sources and effects applied to these sources at various times based on the configuration of the media timeline.

[0038] The media timeline may be configured in a variety of ways to address a variety of considerations. For example, the media timeline may be configured to provide a single timeline model for different multimedia applications, further discussion of which may be found in relation to FIGS. 3-6. In an additional example, the media timelines are configured for storing metadata. Nodes of a media timeline, for instance, may include metadata that describe media referenced by the nodes, such as sources for the media, start and stop times for output of the media, effects to be applied to an output of the media, transitions between media outputs, and so on. Further discussion of media timelines and the metadata therein may be found in relation to FIGS. 7-11.

[0039] In a further example, the media timeline may be configured to support dynamic changes to the media timeline. For example, nodes of the media timeline may be dynamically changed, such as modified, added, and/or removed. Other nodes of the media timeline that are affected by these changes may be automatically updated by the media timeline. Additionally, the media timeline is configured for dynamic creation and/or loading of the media timeline. Further discussion of media timelines that provide for dynamic creation and/or loading of the nodes of the media timeline may be found in relation to FIGS. 12-13.

[0040] In the following discussion, exemplary structures and components of the media timeline are first discussed in relation to FIGS. 2-18. The timeline source which may be utilized to render the exemplary media timelines is described in relation to FIGS. 19-24.

[0041] **Exemplary Environment**

FIG. 1 is an illustration of an environment 100 in an exemplary implementation in which a computer 102 provides access to a plurality of media. The computer 102, as

illustrated, is configured as a personal computer (PC). The computer 102 may also assume a variety of other configurations, such as a mobile station, an entertainment appliance, a set-top box communicatively coupled to a display device, a wireless phone, a video game console, a personal digital assistant (PDA), and so forth. Thus, the computer 102 may range from a full resource device with substantial memory and processor resources (e.g., PCs, television recorders equipped with hard disk) to a low-resource device with limited memory and/or processing resources (e.g., a traditional set-top box). An additional implementation of the computer 102 is described in relation to FIG. 25.

[0042] The computer 102 may obtain a variety of media from a variety of media sources. For example, the computer 102 may locally store a plurality of media 104(1), ..., 104(k), ..., 104(K). The plurality of media 104(1)-104(K) may include an assortment of audio and video content having various formats, such as WMV, WMA, MPEG 1, MPEG 2, MP3, and so on. Further, the media 104(1)-104(K) may be obtained from a variety of sources, such as from an input device, from execution of an application, and so on.

[0043] The computer 102, for instance, may include a plurality of applications 106(1), ..., 106(n), ..., 106(N). One or more of the plurality of applications 106(1)-106(N) may be executed to provide media, such as documents, spreadsheets, video, audio, and so on. Additionally, one or more of the plurality of applications 106(1)-106(N) may be configured to provide media interaction, such as encoding, editing, and/or playback of the media 104(1)-104(K).

[0044] The computer 102 may also include a plurality of input devices 108(1), ..., 108(m), ..., 108(M). One or more of the plurality of input devices 108(1)-108(M) may be configured to provide media for input to the computer 102. Input device 108(1), for

instance, is illustrated as a microphone that is configured to provide an input of audio data, such as a voice of the user, a song at a concert, and so on. The plurality of input devices 108(1)-108(M) may also be configured for interaction by a user to provide inputs that control execution of the plurality of applications 106(1)-106(N). For example, input device 108(1) may be utilized to input voice commands from the user, such as to initiate execution of a particular one of the plurality of applications 106(1)-106(N), control execution of the plurality of applications 106(1)-106(N), and so forth. In another example, input device 108(m) is illustrated as a keyboard that is configured to provide inputs to control the computer 102, such as to adjust the settings of the computer 102.

[0045] Further, the computer 102 may include a plurality of output devices 110(1), ..., 110(j), ..., 110(J). The output devices 110(1)-110(J) may be configured to render media 104(1)-104(K) for output to the user. For instance, output device 110(1) is illustrated as a speaker for rendering audio data. Output device 110(j) is illustrated as a display device, such as a television, that is configured to render audio and/or video data. Thus, one or more of the plurality of media 104(1)-104(K) may be provided by the input devices 108(1)-108(M) and stored locally by the computer 102. Although the plurality of input and output devices 108(1)-108(M), 110(1)-110(J) are illustrated separately, one or more of the input and output devices 108(1)-108(M), 110(1)-110(J) may be combined into a single device, such as a television having buttons for input, a display device, and a speaker.

[0046] The computer 102 may also be configured to communicate over a network 112 to obtain media that is available remotely over the network 112. The network 112 is illustrated as the Internet, and may include a variety of other networks, such as an

intranet, a wired or wireless telephone network, a broadcast network, and other wide area networks. A remote computer 114 is communicatively coupled to the network 112 such that the remote computer 114 may provide media to the computer 102. For example, the remote computer 114 may include one or more applications and a video camera 116 that provides media, such as home movies. The remote computer 114 may also include an output device to output media, such as the display device 118 as illustrated. The media obtained by the computer 102 from the remote computer 114 over the network 112 may be stored locally with the media 104(1)-104(K). In other words, media 104(1)-104(K) may include locally stored copies of media obtained from the remote computer 114 over the network 112.

[0047] Thus, the computer 102 may obtain and store a plurality of media 104(1)-104(K) that may be provided both locally (e.g., through execution of the plurality of applications 106(1)-106(N) and/or use of the plurality of input device 108(1)-108(M)), and remotely from the remote computer 114 (e.g., through execution of application and/or use of input devices). Although the plurality of media 104(1)-104(K) has been described as stored on the computer 102, the media 104(1)-104(K) may also be provided in "real-time". For example, audio data may be streamed from the input device 108(1), which is illustrated as a microphone, without storing the audio data.

[0048] The computer 102 includes a timeline generator 120 that, when executed on the computer 102, generates a media timeline 122. For example, the timeline generator 120 may be configured as an application that exposes one or more software components that may be used to generate the media timeline 122, such as through a user interface by a user. As previously described, the media timeline 122 provides a technique for a user to

define a presentation of stored and/or real-time media from the plurality of media sources. For example, the media timeline 122 may describe a collection of media that was obtained from the input devices 108(1)-108(M), the applications 106(1)-106(N), and/or the remote computer 114. The user may utilize one or more of the input devices 108(1)-108(M) to interact with the timeline generator 120 to define groupings and/or combinations of the media 104(1)-104(K). The user may also define an order and effects for presentation of the media 104(1)-104(K). A timeline source 124 may then be executed on the computer 102 to render the media timeline 122. The media timeline 122, when rendered, provides the expressed groupings and/or combinations of the media 104(1)-104(K) for rendering by one or more of the plurality of output devices 110(1)-110(J). Additionally, the timeline generator 120 may also programmatically generate the media timeline 122 as is described in greater detail in the following implementation.

[0049] FIG. 2 is a high level block diagram of a system 200 in an exemplary implementation in which the system 200, implemented in software, includes an application 202 that interacts with a media foundation 204 to control presentation of a plurality of media 206(g), where "g" can be any number from one to "G". The media foundation 204 may be included as a part of an operating system to provide playback of the media 206(g) such that applications that interact with the operating system may control playback of the media 206(g) without "knowing" the particular details of the media formats. The media 206(g) may be provided from a variety of sources, such as from the media 104(1)-104(K) of FIG. 1, through execution of the applications 106(1)-106(N), use of the input devices 108(1)-108(M), output devices 110(1)-110(J), and so on.

[0050] The application 202, which may be the same as or different from applications 106(1)-106(N) of FIG. 1, interacts with a media engine 208 to control the media 104(1)-104(K). In at least some embodiments, the media engine 208 serves as a central focal point of the application 202 that desires to somehow participate in a *presentation*. A presentation, as used in this document, refers to or describes the handling of media. In the illustrated and described embodiment, a presentation is used to describe the format of the data on which the media engine 208 is to perform an operation. Thus, a presentation can result in visually and/or audibly presenting media, such as a multimedia presentation in which both audio and accompanying video is presented to user within a window rendered on a display device, such as output device 110(j) of FIG. 1 that is illustrated as a display device that may be associated with a desk-top PC. A presentation can also result in writing media content to a computer-readable medium such as a disk file. Thus, a presentation is not limited to scenarios in which multimedia content is rendered on a computer. In some embodiments, operations such as decoding, encoding and various transforms (such as transitions, effects and the like), can take place as a result of a presentation.

[0051] In an embodiment, the media foundation 204 exposes one or more application program interfaces that can be called by the application 202 to interact with the media 206(g). For example, the media foundation 204 may be thought of as existing at an "infrastructure" level of software that is executed on the computer 102 of FIG. 1. In other words, the media foundation 204 is a software layer used by the application 202 to interact with the media 206(g). The media foundation 204 may be utilized to control a number of aspects of the media 206(g), such as output, rendering, storage, and so on.

Thus, the media foundation 204 may be utilized such that each application 202 does not have to implement separate code for each type of media 206(g) that may be used in the system 200. In this way, the media foundation 204 provides a set of reusable software components to do media specific tasks.

[0052] The media foundation 202 may utilize several components among which include the media timeline 122, the timeline source 124, a media source 210, a media processor 212, a media session 214, the media engine 208, a source resolver 216, one or more transforms 218, one or more media sinks 220, 222, and so on. One advantage of various illustrated and described embodiments is that the system 200 is a pluggable model in the sense that a variety of different kinds of components can be utilized in connection with the systems described herein. Also included as a part of system 200 is a destination 224, which is discussed in more detail below. In at least one embodiment, however, the destination 224 is an object that defines where a presentation is to be presented (e.g. a window, disk file, and the like) and what happens to the presentation. That is, the destination may correspond to one or more of the media sinks 220, 222 into which data flows.

[0053] The media timeline 122 employs a timeline object model which provides a way for a user to define a presentation based on media that is rendered by the timeline source 124. The media timeline 122 may range from a sequential list of media files to more complex forms. For example, the media timeline 122 may employ file structures, such as SMIL and AAF, to express media playback experiences that include transitions between media, effects, and so on. The application 202, for instance, may be configured as a media player that can play a list of songs, which is commonly referred to as a playlist. As

another example, in an editing system a user may overlay one video over the other, clip a media, add effect to the media and so forth. Such groupings or combinations of media may be expressed using the media timeline 122. Further discussion of the media timeline 122 is found in relation to FIG. 3.

[0054] The media source 210 is utilized to abstract a provider of media. The media source 210, for instance, may be configured to read a particular type of media from a particular source. For example, one type of media source might capture video from the outside world (a camera), and another might capture audio (a microphone). Alternately or additionally, the media source 210 may read a compressed data stream from disk and separate the data stream into its compressed video and compressed audio components. Yet another media source 210 might obtain data from the network 112 of FIG. 1. Thus, the media source 210 may be utilized to provide a consistent interface to acquire media.

[0055] The media source 210 provides one or more media presentation 226 objects (media presentation). The media presentation 226 abstracts a description of a related set of media streams. For example, the media presentation 226 may provide a paired audio and video stream for a movie. Additionally, the media presentation 226 may describe the configuration of the media source 210 at a given point in time. The media presentation 226, for instance, may contain information about the media source 210 including descriptions of the available streams of the media source 210 and their media types, e.g. audio, video, MPEG, and so on.

[0056] The media source 210 may also provide a media stream 228 object (media stream) which may represent a single stream from the media source 210 which can be accessed by the application 202, i.e. exposed to the application 202. The media stream

228 thus allows the application 202 to retrieve samples of the media 206(g). In an implementation, the media stream 228 is configured to provide a single coherent stream of data. For instance, in a WMV file, there may be one media stream for video data and another media stream for audio data and therefore two media streams object may be employed to provide the respective streams.

[0057] In the media foundation 204, therefore, the media source 210 is defined as a software component which outputs samples for a presentation. The timeline source 124 interprets the media timeline 122, but at the same time, may also act in a manner similar to the media source 210. For example, the timeline source 210 may be utilized to hide the intricacies of rendering the media timeline 122 to provide media described by the media timeline 122 from other components of the media foundation 204.

[0058] The media processor 212 manages data flow in a topology 230. The topology 230 defines how data flows through various components for a given presentation. A "full" topology includes each of the components, e.g. software modules, used to manipulate the data such that the data flows with the correct format conversions between different components. When a topology is created, the user might choose to create it partially. This partial topology is not sufficient, by itself, to provide a final presentation. Therefore, a component called the topology loader 232 may take the partial topology and convert it into a full topology by adding the appropriate data conversion transforms between the components in the partial topology.

[0059] In the topology 230, for example, data generally originates at the media source 210, flows through one or more transforms 218, and proceeds into one or more media sinks 220, 222. Transforms 218 can include any suitable data handling components that

are typically used in presentations. Such components can include those that uncompress compressed data and/or operate on data in some way, such as by imparting an effect to the data, as will be appreciated by the skilled artisan. For example, for video data, transforms can include those that affect brightness, color conversion, and resizing. For audio data, transforms can include those that affect reverberation and re-sampling. Additionally, decoding and encoding can be considered as transforms.

[0060] Media sinks 220, 222 are typically associated with a particular type of media content. Thus, audio content might have an associated audio sink such as an audio renderer. Likewise, video content might have an associated video sink such as a video renderer. Additional media sinks can send data to such things as computer-readable media, e.g. a disk file and the like.

[0061] The media session 214 is a component which may schedule multiple presentations. Therefore, the media processor 212 may be used to drive a given presentation, and the media session 214 utilized to schedule multiple presentations. The media session 214, for instance, may change topologies that are rendered by the media processor 212. For example, the media session 214 may change from a first topology that is rendered on the media processor 212 to a second topology such that there is no gap between the renderings of samples from the consecutive presentations that are described by the respective topologies. Thus, the media session 214 may provide a seamless user experience as the playback of the media moves from one presentation to another.

[0062] The source resolver 216 component may be utilized to create a media source 210 from URLs and/or byte stream objects. The source resolver 216 may provide both

synchronous and asynchronous ways of creating the media source 210 without requiring prior knowledge about the form of data product by the specified resource.

[0063] In at least one embodiment, the media foundation 204 is utilized to abstract away the specific details of the existence of and interactions between various components of the media foundation 204. That is, in some embodiments, the components that are seen to reside inside the media foundation 204 are not visible, in a programmatic sense, to the application 202. This permits the media foundation 202 to execute so-called "black box" sessions. For example, the media engine 208 can interact with the media session 214 by providing the media session certain data, such as information associated with the media (e.g. a URL) and the destination 224, and can forward the application's 202 commands (e.g. open, start, stop and the like) to the media session 214. The media session 214 then takes the provided information and creates an appropriate presentation using the appropriate destination.

[0064] The media foundation 204 may also include a timeline plugin 234. The timeline plugin 234 may be utilized such that different media timeline file formats may be "plugged-in" to the media foundation 204. For example, a bytestream plugin 236 may be written for a format in question and registered with the media foundation 204. The source resolver 216 may then invoke a bytestream plugin 236 when a file of that type is opened. In turn the bytestream plugin 236 can parse the file, create a media timeline 122 representing the presentation described in the file, and create a timeline source 124 for it. In general, the bytestream plugin 236 is responsible for reading the raw bytestream and creating a media source 208 for it. In an implementation, the remaining components of media foundation 204 are not made aware that the media source created in this instance is

a timeline source 124. Therefore, the timeline source 124 is treated like any other media source 208. In an implementation, a bytestream plugin 236 that can parse a media timeline 122 and create a timeline source 124 is referred to as a timeline plugin, which is described in greater detail in relation to FIG. 23.

[0065] The timeline plugin 234 may also provide an interface such that the application 202 may interact with the timeline plugin directly, such as to load and save the media timeline 122 from or to a file. For example, the timeline plugin 234 may be created and then called to initiate a load function to provide a bytestream. The timeline plugin 234 may then parse the file and create a root node and any additional nodes to create the media timeline 122, which will be described in greater detail in relation to FIG. 3. The timeline plugin 234 may also be used to persist the media timeline 122 to different formats. For example, the application 202 may create the media timeline 122 programmatically. In other words, the application may act as the timeline generator 120 of FIG. 1. The application 202 may then create a timeline plugin for ASX files, and ask the timeline plugin to save the media timeline 122 in the ASX format. In another example, a user can open an m3u file, i.e. a playlist file format for specifying multiple MP3 files, get the media timeline 122 from it, and then ask the timeline plugin to save the media timeline 122 in the ASX format. In this example, the timeline plugin acts as the timeline generator 120. Thus, the media foundation 204 may expose a plurality of software components that provide media functionality over an application programming interface for use by the application 202.

[0066] FIG. 3 is an illustration of an exemplary implementation in which a media timeline 300 is shown as a tree that includes a plurality of nodes that describe an output

of media for a presentation. The media timeline 300, which may or may not correspond to the media timeline 122 of FIGS. 1 and 2, is structured as a tree that includes a plurality of nodes 302-312. Each of the plurality of nodes 302-312 includes respective metadata 314-322 that describes various attributes and behaviors for the node and/or "children" of that particular node. For example, node 304 and node 306 are arranged, respectively, as a "parent" and "child". Node 304 includes metadata 316 that describes behaviors and attributes of that node 304. The metadata 316 may also describe each of the "child" nodes 306, 308, such as a rendering order of the nodes 306, 308.

[0067] In an implementation, the media timeline 300 is not executable by itself to make decisions about a user interface (UI), playback or editing. Instead, the metadata 314-324 on the media timeline 300 is interpreted by a software and/or hardware component that renders the media timeline 300, such as the timeline source 124 of FIG. 2. Additionally, applications that are utilized during rendering of the media timeline 122 may obtain relevant metadata for that particular application. For example, the application 202 of FIG. 2 may be configured as a playback engine that is only interested in the times at which each media referenced in the media timeline is to be started. On the other hand, another application, such as a media player, may be interested in just displaying the titles of the songs, which are stored as metadata on each node. In this way, the metadata may be utilized at the same time by one or more applications that utilize an output of the media.

[0068] The nodes 302-312, as positioned on the media timeline 300, describe a basic layout of the media timeline 300. This layout may be utilized for displaying a timeline structure in a user interface, utilized by the timeline source 124 of FIG. 2 to order

rendering of the nodes, and so forth. For instance, various types of nodes 302-312 may be provided such that a desired layout is achieved. The node type indicates how the children of that node are interpreted, such as a root node 302 and leaf nodes 308-312. The root node 302 specifies a starting point for rendering the metadata timeline 300 and includes metadata 314 that describes how rendering is to be initiated.

[0069] In the illustrated implementation of FIG. 3, the leaf nodes 308, 310, 312 of the media timeline 122 directly map to media. For example, the leaf nodes 308, 310, 312 may have respective metadata 320, 322, 324 that describes how to retrieve the media that each of the leaf nodes 308-312 represent. A leaf node may specify a path for an audio and/or video file, point to a component which generates video frames programmatically during rendering of the media timeline 300, and so on. Leaf node 308, for instance, includes metadata 320 having a pointer 326 that maps to input device 108(1) that is configured as a microphone. Leaf node 310 includes metadata 322 having a pointer 328 that maps to an address of the media 330 in a storage device 332 that is included locally on the computer 102 of FIG. 1. Leaf node 312 includes metadata 324 having a pointer 334 that maps to a network address of the remote computer 114 on the network 112. The remote computer 114 includes the video camera 116 to provide media over the network 112 to the computer 102 of FIG. 1. Thus, in this implementation, the timeline 300 does not include the actual media, but rather references the media by using pointers 326, 328, 334 that describe where and/or how to locate the referenced media.

[0070] Nodes 304, 306 may also describe additional nodes of the media timeline 300. For example, node 304 may be utilized to describe the order of execution for nodes 306, 308. In other words, node 304 acts as a "junction-type" node to provide ordering and

further description of its "children". There are a variety of junction-type nodes that may be utilized in the media timeline 300, such as a sequence node and a parallel node. FIGS. 4-6 describe exemplary semantics behind the sequential and parallel nodes.

[0071] FIG. 4 is an illustration of an exemplary implementation 400 in which a sequence node 402 and a plurality of leaf nodes 404, 406, 408 that are children of the sequence node 402 are shown. The children of the sequence node 402 are rendered one after the other. Additionally, the sequence node 402 may include metadata 410 that describes a rendering order of the plurality of leaf nodes 404-408. As illustrated, leaf node 404 is rendered first, followed by leaf node 406, which is followed by leaf node 408. Each leaf node 404-408 includes respective metadata 412, 414, 416 having respective pointers 418, 420, 422 to respective media 424, 426, 428. Thus, the sequence node 402 may represent the functionality of a linear playlist of files.

[0072] Although the child nodes of the sequence node 402 are configured as leaf nodes in this implementation, child nodes of the sequence node 402 may represent any other type of node. For example, child nodes may be utilized to provide a complex tree structure as shown in FIG. 3. Node 306 of FIG. 3, for instance, is the child of another junction-type node, i.e. node 304.

[0073] FIG. 5 is an illustration of an exemplary implementation 500 in which a sequence node 502 and a plurality of nodes 504, 506, that are children of the sequence node 502 include metadata that specifies timing information for execution of the respective plurality of nodes 504, 506. Each of the plurality of leaf nodes 504, 506, 508 includes respective metadata 510, 512, 514 as previously described. The metadata 510 of leaf node 504, which is a first child node of the sequence specified by node 502, includes time

516 data that specifies a start time relative to the start time on the parent node, i.e. node 502. The other child nodes, i.e. leaf nodes 506, 508 have their start times specified relative to the previous node in the sequence. For example, suppose output of media corresponding to the leaf node 504 is desired when the node 502 is first executed. Additionally, when output of the media referenced by leaf node 504 ends, the media corresponding to leaf node 506 is to be output after a gap to 20 seconds. Therefore, the time 516 specified by the metadata 510 of leaf node 504 is "zero" and the time 518 specified by the metadata 512 of leaf node 506 is "20 seconds".

[0074] Specifying times 516, 518 relative to the previous node allows for defining a sequence where duration output of media referenced by each child node in the sequence is not known. When the start time for a node is not specified, as shown by the metadata 514 of leaf node 508, it means that the node, i.e. leaf node 508, should be immediately start output after the previous node, i.e. leaf node 506, has finished output.

[0075] FIG. 6 is an illustration of an exemplary implementation 600 in which a parallel node 602 includes metadata 604 specifying a plurality of leaf nodes 606, 608 that are children of the parallel node 602 are shown. In the previous implementations that were described in relation to FIGS. 4 and 5, sequence nodes were discussed in which nodes that are children of the sequence node were rendered, one after another. To provide rendering of nodes at the same time, the parallel node 602 may be employed.

[0076] The children of the parallel node 602 may be rendered simultaneously. For example, leaf node 606 and leaf node 608 are children of parallel node 602. Each of the leaf nodes 606, 608 includes respective metadata 610, 612 having respective pointers 614, 616 to respective media 618, 620. Each of the leaf nodes 606, 608 includes a

respective time 622, 624 included in the respective metadata 610, 612 that specifies when the respective leaf nodes 606, 608 are to be rendered. The times 622, 624 on the leaf nodes 606, 608 are relative to the parallel node 602, i.e. the parent node. Each of the child nodes can represent any other type of node and combinations of nodes, providing for a complex tree structure with combined functionality. For example, a "junction" type node may also reference media, and so forth. Although metadata including time data has been described, a variety of metadata may be included on nodes of the media timeline, an example of which is described in the following implementation.

[0077] Media Timelines that are Configured to Store Metadata

FIG. 7 is an illustration of an exemplary implementation 700 showing examples of metadata that may be included in a node. In this implementation 700, each node in the media timeline is capable of storing metadata. Metadata may be utilized to describe properties of the node, such as media referenced by the node, rendering order for children of the node, specify a node type (e.g., sequential, parallel, root, and leaf), and so forth. The media timeline may treat any property being set on the node as metadata. For example, properties like start and stop times of the node, the URL for the media of the node, and so on are stored as metadata.

[0078] Additionally, authors of the media timeline may add custom metadata to the nodes. For example, the application 202 of FIG. 2 may be configured as a media player that stores album art for a CD track on the leaf node corresponding to that particular track. Standard properties and custom properties may be treated in the same manner so that there is no ambiguity when obtaining the metadata. Therefore, even if each property

described by the metadata is provided by a different respective interface or source, the media timeline provides a mechanism to track the various properties.

[0079] Further, properties from different sources may be aggregated by treating the metadata in a consistent manner by the media timeline. For example, a playlist may include a plurality of tracks, each having a different composer. Each track of the playlist may be represented as a leaf node that is a child of a sequence node. The media timeline may aggregate the metadata such that a query to the sequence node, i.e. the parent node, returns the composers of all the media in the playlist from each leaf node, i.e. the child nodes. Consistent use of metadata may also provide sorting for each of the nodes. For example, if all properties on a node are treated as metadata, an application may sort the nodes based on any properties defined in the metadata in a consistent fashion.

[0080] A node 702 may include a variety of metadata 704, such as properties that define playback behaviors and attributes for the nodes. Examples of properties defined by the metadata 704 are described as follows.

[0081] URL 706

This property holds the URL for the media. In the case of a file, the URL 706 property may provide the path to the file. For example, the URL 706 property may provide a path to a storage device to locate particular media.

[0082] SourceObject 708, SourceObjectID 710

In some instances, the source for the media cannot be specified by a URL. For example, a media source for outputting black color frames may not be locatable by a URL. The SourceObject 708 and SourceObjectID 710 properties allow the user to specify the media source by specifying an object which can resolve to a media source,

such as the media source itself or some other object. When a media source is specified as a source object, SourceObject 708 property provides a pointer to the media source and the SourceObjectID 710 property specifies a globally unique identifier of the source object. In an implementation, the SourceObject 708 property takes precedence over the URL 706 property in case both are defined.

[0083] Start Time 712, Stop Time 714

The start and stop times 712, 714 define at what time the node 702 is to be started and stopped with respect to the other nodes. For nodes that are children of a parallel node, for instance, the start and stop times 712, 714 are defined relative to the parallel node, i.e. the parent of the children. For nodes that are children of a sequence node, the first child node includes start and stop times 712, 714 that are defined relative to the sequence node. The remaining nodes each include start and stop times that are defined relative to a previous sibling. In an implementation, it is not necessary to define the start and stop times 712, 714 for the node 702. For example, when the start and stop times 712, 714 are not specified, the start time 712 is assumed to be zero and the node 702 is stopped when the rendering of the media referenced by the node 702 is completed.

[0084] Media Start 716, Media Stop 718

Each node in a media timeline may reference media. The media start 716 and media stop 718 properties define a portion of the media that is to be output. For example, the node 702 may represent media from a file having a total length of 50 seconds. The user, however, might want to output only a portion of the media from 20 to 30 seconds in the file. To do this, the media start 716 may be specified as 20 seconds and the media stop 718 may be specified as 30 seconds.

[0085] The duration of the time period defined by the start time 712 and stop time 714 of the node, i.e. "nodetime" need not equal the duration of the time period defined by the media start 716 and the media stop 718, i.e. "mediatime". For example, when the specified nodetime is greater than the mediatime, output of the media referenced by the node 702 may be slowed. Therefore, the portion of the media defined by the media start 716 and the media stop 718 may be output for the duration of the time period defined by the start and stop times 712, 714 of the node, i.e. "nodetime". In other words, output of the portion may be extended such that the nodetime is equal to the mediatime. In another example, a last frame of the media may be frozen until the nodetime elapses, a video frame can be made blank (e.g., black), and so on. Similarly, if the nodetime is less than the mediatime, the media may be output at a faster rate such that output is finished within the specified nodetime. In a further example, output of the media may be truncated. For instance, any portion of the segment defined by the mediatime that is greater than the nodetime is not output. In an implementation, the media timeline itself does not enforce these behaviors, but rather these behaviors are read by the timeline source 124 when rendering the media timeline 122 as described in relation to FIG. 1.

[0086] When the media stop 718 for the node 702 is not specified, the media referenced by the node 702 is output until completion. For example, in a player scenario, a user may desire the output of a playlist of media that does not have the duration of each media item referenced. Additionally, "back to back" output of the media included in the playlist may be desired. To represent this case on the media timeline, a sequence node may be created having leaf nodes that are children of the sequence node which do not have a specified media stop 718 properties.

[0087] Time Format 720

The time-based properties described previously may have an accompanying time format 720 property (time format). Examples of time formats include 100 nanosecond units, frame number, time code, and so on. Thus, the time format 720 may specify the time format for the start time 712, stop time 714, media start 716 and media stop 718. Additionally, the time format 720 may specify different formats for each of the time-based properties. For instance, the start and stop times 712, 714 may utilize a time format of 100 nanosecond units, while the media start 716 and media stop 718 time formats may utilize frame counts.

[0088] Stream Selection 722

The stream selection 722 property can be utilized on the node 702 in a variety of ways. For example, the stream selection 722 property may act as a filter such that media having desired characteristics is provided. The node 702, for instance, may reference both audio and video streams of media, such as a television program. The user, however, may only be interested in only the video stream, even if the URL 706 specified on the node 702 points to both the audio and video streams. In such a case, the audio stream from the media is not exposed, such that it appears to the user that the node 702 provides only video media.

[0089] Format Based 724

Format based 724 properties may be utilized to specify other properties such as frame rate, pixel aspect ratio, audio sampling rate, and so on, that are desired from the node 702. The appropriate transforms for converting to/from these formats are then inserted into the rendered media timeline during playback.

[0090] Loop Count 726

The loop count 726 property may be used to specify how many times the rendering of the node 726 is to be repeated. For example, if the loop count 726 property is negative, the output of the media referenced by the node 702 may be repeated infinitely.

[0091] Disabled 728

The node 702 may be disabled by setting the disabled 728 property. For example, if the disabled 728 property is set to "true", the node 702 is ignored during rendering of the media timeline. For instance, a sequence of three leaf nodes may be provided in a media timeline. If the second node in the media timeline is disabled, i.e. the disabled 728 property is set to "true", output of the media referenced by the media timeline will appear as if the media timeline has only the first and third nodes.

[0092] NoSkip 730

The NoSkip 730 property is a feature which can be used by timeline authors to specify media which cannot be skipped during rendering of the media timeline. When the node 702 is specified as a NoSkip node, i.e. the NoSkip property is set to "true", the user cannot skip to another node after the specified node 702, and cannot fast forward the media being output as part of that node 702. The user, however, may skip to any node "before" that node 702. In another implementation, if the NoSkip 730 property is specified on a parent node, the user will not be able to skip any of the children in the subtree of that node. In a further implementation, the NoSkip 730 property applies only to a sequence node and its immediate children, e.g. children of the sequence node that directly follow the sequence node instead of being included in a another sequence node

that is a child of that sequence node, and is not specified for a parallel node or its immediate children. For example, the NoSkip 730 property may be used to prevent the skipping of advertisements referenced by leaf nodes that are children of a first sequence node. A second sequence node may also be a child of the first sequence node, and include leaf nodes that reference media that can be skipped, such as a television program.

[0093] The NoSkip 730 property may also be utilized to define collections of nodes through which a user may navigate. For example, a media timeline may include a sequence of ten leaf nodes, with the third and seventh nodes being NoSkip nodes, i.e. the NoSkip property is set as "true". Therefore, the user may skip the rendering of the first and second leaf nodes, but cannot skip to the fourth, fifth, sixth, seventh, eighth, ninth, or tenth nodes. Similarly during the rendering of the media timeline from node four to node seven, the user may skip to any node below the seventh node, but may not skip to a node "above" the seventh node, i.e. the eighth, ninth and tenth nodes.

[0094] NoSkip Child 732

Media timelines may support sparse children, i.e. all nodes are not loaded and/or created on the media timeline when the media timeline is initially loaded. Therefore, the children may be loaded and/or created as needed. Further discussion of dynamic loading and creation of nodes may be found in relation to FIGS. 12 and 13. When loading the nodes in a media timeline in this instance, parent nodes may be loaded which have child nodes that are specified as "NoSkip". To indicate that there is the NoSkip 730 property for a child node, the NoSkip child 732 property for the parent node may be used.

[0095] The NoSkip child 732 property may be set at a parent node to indicate whether the parent node includes a child node having the NoSkip 730 property set as "true".

During the rendering of the media timeline, the NoSkip child 732 is used to indicate that all the previous siblings of a node should be checked to determine if navigation to the node is valid. NoSkip child 732 may also be set on a parallel node. For example, if any node in a subtree of the parallel node has the NoSkip 730 property set as "true". In this way, navigation between nodes may be provided that protects the use of the NoSkip 730 property.

[0096] When a node with the NoSkip 730 property set as "true" is added to the media timeline, the media timeline may automatically set the NoSkip Child 732 property as "true" on all the parents of the added node. This way a rendering engine, e.g. timeline source 124 of FIG. 1, can optimize which nodes of the media timeline to load and check to determine if the NoSkip 730 property is set as "true".

[0097] **Timeline Effects**

Timeline effects allow the author of a media timeline to specify components which analyze and/or change the appearance of the media. For example, the author might want to show a video in black & white, add echo to an audio file, show one video on top of another (e.g., picture in picture), and so on. In an implementation, an effect is not a separate node by itself. To provide the effect for the media, the author may specify effects in the metadata in the node. For example, the metadata may include an array of effects that are defined on the node. The array may specify a series of effects to be applied to the output of that node, i.e. when the media referenced by the node is rendered. In this implementation, the effect is not an object which actually implements the effect, but rather specifies properties and attributes which describe how to create and apply the effect. This is similar to how the node references the media in the previous

implementations. For example, as discussed in relation to FIG. 3, the leaf nodes 308-312 themselves do no contain the media, but rather include respective metadata 320-324 having respective pointers 326, 328, 332 which specify how to obtain the media. The component which actually implements the effect is loaded at runtime by the timeline source that executes the media timeline. Although metadata that includes effect has been described, the effects may also be specified separately. Additionally, in another implementation, the effect is provided by an object in the media timeline that implements the effect.

[0098] Effects specified on nodes of a media timeline may have times that are specified relative to the start time of that node. For example, an effect may be specified on a leaf node that has a start time of ten seconds. Therefore, the effect will be applied to the node, when rendered, after that node has begun output and ten seconds have elapsed.

[0099] Multiple effects can be specified on a node. Additionally, the author of the media timeline may also control the order in which these effects are applied. For example, the author may set a priority on the effect. There are a variety of effects that may be specified by a node. Examples of effects that can be specified on the media timeline include: (1) a simple effect; (2) a composite effect; and (3) a transition effect. Further discussion of these exemplary effects may be found in relation to FIGS. 8-10.

[00100] Simple Effect

A simple effect represents a component which receives a single stream of audio/video and outputs another stream. In other words, it is a one-in/one-out component. For example, an echo effect may receive an audio stream and output a modified audio stream that echoes, provide a "black and white" effect in which video is

shown as black and white, an age effect in which video is made to appear as if it was captured several decades ago, and so on.

[00101] FIG. 8 is an illustration of an exemplary implementation 800 in which a node 802 includes metadata 804 that specifies a plurality of simple effects 806, 808, 810. As shown in FIG. 8, the plurality of simple effects 806-810 may be applied to media 812 included in a storage device 814 that is referenced by a pointer 816. In this example, the plurality of effects 806-810 are concatenated, i.e. the output for the first effect 806 goes to the input of the second effect 808 and so on, to provide the multiple effects to the media 812. Each of the plurality of effects 806-810 may be given a priority to provide an order for processing the effects. For instance, the priorities of the effects may determine the order the effects are concatenated. If there is a contention on the priority, the effects may be added in the order each effect was specified in the effect array.

[00102] In an implementation, the duration of the plurality of effects 806-810 does not change the duration of the media 812. For example, the processing of the plurality of effects 806-810 may be truncated at the time boundaries of the node 802. For instance, the rendering of the media 812 may have a duration of 10 seconds. The processing of the plurality of effects 806-810, however, may have a duration of 20 seconds. In such an instance, the timeline source 124 of FIG. 1 may finish processing of the plurality of effects 806-810 for node 802 at 10 seconds.

[00103] When defining the effects 806-810, the author of the timeline may explicitly specify the inputs and the outputs of each of the effects 806-810. For example, each of the effects 806-810 may include data that describes which stream is connected to which effect input. Each of the effects 806-810 may also have respective data that describes the

major type of the respective effect's 806-810 output, e.g. audio, video, and so on. Further, each of the effects 806-810 may include metadata that describes a start time and/or a stop time of the effect within the node.

[00104] Composite Effects

A composite effect may be used to process media of the children of a parallel node to give a resultant output. For example, FIG. 9 is an illustration of an exemplary implementation 900 showing a parallel node 902 that provides a composite effect to the outputs of two or more child nodes. Parallel node 902 in this implementation is similar to the parallel node 602 that was described in relation to FIG. 6.

[00105] Parallel node 902 includes an array of composite effects 906, 908, 910. When specifying a composite effect, the author of the media timeline specifies how to connect the inputs of the effects 906-910 and also the major types for the outputs from the effects 906-910. For example, leaf node 912 and leaf node 914 may be configured as the children of the parallel node 902. As previously described, each leaf node 912, 914 includes respective metadata 916, 918 having respective pointers 920, 922 that reference respective media 924, 926. The leaf nodes 912, 914, when rendered, provide media 924, 926 for output.

[00106] The effects 906, 908, 910 are applied to the output of the media 924, 926 that are specified by the parallel node 902. For example, the parallel node 902 may provide a rotating cube with a different media (e.g., video) on each face of the cube, a scrolling roll of film with different media playing in each frame of the film, and so forth.

[00107] Although parallel node 902 was described as applying the plurality of effects 906-910 to each of the leaf nodes 912, 914, in additional implementations the parallel node

902 might apply the effects 906-910 to only a few of the children of the parallel node 902. In other words, the effects 906-910 need not be applied to all of the nodes that are children of the parallel node 902. For example, the metadata 904 and/or effects 906-910 may specify one or more particular nodes to apply one or more of the plurality of effects 906-910.

[00108] Transition Effect

FIG. 10 is an illustration of an exemplary implementation 1000 in which a transition effect is specified to supply an effect between an output of media referenced by a previous node to an output of media referenced by a subsequent node. In FIG. 10, a sequence node 1002 is shown that is similar to sequence node 402 that was described in relation to FIG. 4. The sequence node 1002 has a plurality of leaf nodes 1004, 1006, 1008 that are children of the sequence node 1002. Each of the plurality of leaf nodes 1004-1008 includes respective metadata 1014-1018 having respective pointers 1020-1024 that reference respective media 1026-1030.

[00109] The sequence node 1002 include metadata 1010 that describes a transition effect 1012 that is to be employed between output of the media 1026-1030 referenced by the respective leaf nodes 1004-1008. Thus, the transition effect 1012 is applied to the media 1026-1030 originating from the children of the sequence node 1002. The transition effect 1012 is utilized to combine two or more media 1026-1038 into a single output. Additionally, the transition effect 1012 may include data that specifies one or more of leaf nodes 1004-1008 to which the transition effect is to be applied. For example, the data may specify that the transition effect 1012 is to be employed between the output of media 1026, 1028. The first input to the transition effect 1012 is supplied by the node for

which it is defined, i.e. leaf node 1004. The next input to the transition effect 1012 is the next node in the sequence, i.e. leaf node 1006. Example of transition effects include an audio cross fade between two nodes that are output in sequence, a "swipe" of a first video with a second video, and so on.

[00110] The transition effect 1012 has a duration 1032. The duration 1032 may be used to specify an amount of overlap desired between the two or more nodes in a sequence. For example, the second input in the sequence, i.e. media 1026, may be output such that it overlaps for the duration 1032 of the transition effect 1012. Hence, an output duration of the sequence node 1002 becomes a function of the times specified on the leaf nodes 1004-1008 and the overlap specified by the duration 1032 of the transition effect 1012.

[00111] Global effects may also be specified. For example, the transition effect 1012 may specify a global transition for each of the children of that node, e.g. leaf nodes 1004-1008 of sequence node 1002. Therefore, if the author of a media timeline desires the use of the same transition for all the leaf nodes 1004-1008, the author may do so by specifying the transition effect 1012 as a global transition. Thus, by specifying a global transition, the author need not specify a separate transition for each node.

[00112] Effect Metadata

FIG. 11 is an illustration of an exemplary implementation 1100 showing a node 1102 having metadata 1104 that includes a plurality of effect metadata 1106. Like the metadata 704 specified on the node 702 as described in relation to FIG. 7, the effect metadata 1106 may also specify a variety of properties for an effect. For example, the effect metadata 1106 may define standard properties which have a specific meaning for the media timeline. The effect metadata 1106 may be used to configure transform objects

which are utilized to provide the effect. Transform objects, for instance, may publish their own set of properties, which are used to configure the transform object to provide the effect. For example, for a color converter transform object there are specific properties for controlling the hue, saturation and brightness. In addition to these properties, other custom properties of interest may be specified for the effect. The following is a listing of examples of effect metadata 1106 that is supported by the media timeline.

[00113] Effect Object GUID 1108

Similar to how nodes may reference media, an effect may reference a transform object that provides the effect. The effect object GUID 1108 property specifies the GUID to be used to create the transform object that provides the effect. For example, during output of the media, the transform object referenced by the effect object GUID 1108 may be created when needed to provide the effect.

[00114] Effect Object 1110

The node 1102 may utilize the effect object 1110 property as a pointer to reference an effect object that provides the effect. The referenced effect object may be used directly during output of the media of the node 1102. The effect object 1110 property takes precedence over the effect GUID, if both are specified.

[00115] Priority 1112

As previously described, when effects are concatenated together, the priority 1112 property may be used to specify the ordering of the effects. If there is more than one effect with the same priority, the effects are applied in the order in which the effects were added to the node 1102.

[00116] Start Time 1114, Stop Time 1116

The start and stop times 1114, 1116 are specified relative to the node 1102 on which the effect is specified. The start and stop times 1114, 1116 define the time at which the effect will be active. If these properties are not specified, the effect will be applied for the entire duration of the output of the media referenced by the node 1102. These properties can be applied to both simple effects that were described in relation to FIG. 8 and composite effects that were described in relation to FIG. 9.

[00117] Time Format 1118

The start and stop times 1114, 1116 may be specified in a variety of formats. The time format 1118 property may be used to specify the format of these time values. A variety of time formats may be utilized, such as 100 nano-second units, frame numbers, time codes, and so on.

[00118] Duration 1120

As previously described in relation to FIG. 10, the duration 1120 property may be used to specify the duration of a transition between the output of respective media. For example, the duration 1120 may be used to specify an amount of overlap between the output of media referenced by two consecutive nodes.

[00119] Number of Inputs 1122, Number of Outputs 1124

Simple effects utilize one input and one output, and therefore the number of inputs and outputs 1122, 1124 may be set automatically in the media timeline for simple effects. A transition effect may employ two inputs and one output. Therefore, the number of inputs and outputs 1122, 1124 may also be set automatically in the media timeline for transition effects. For composite effects, an author may define as many

inputs and/or outputs as desired. Therefore, the number of inputs and outputs 1122, 1124 may be set by the author to reflect the number of inputs and outputs for the transform object that provides the effect.

[00120] Output Major Type 1126

The output major type 1126 is specified for each output of the effect. Specifying output major type 1126 property facilitates connecting the effect to other effects or destinations. For example, the author of a media timeline may readily determine the major type, i.e. audio, video, and so on, of the output and therefore efficiently specify connections between relevant effects, e.g. audio effect to audio effect.

[00121] Input Connections 1128

Once the effect has been defined, the author may specify media that is to be processed by the effect. The input connections 1128 property may be used to identify the media to be connected to each of the effect inputs.

[00122] **Dynamic Creation and Loading of Nodes of a Media Timeline**

Dynamic creation and loading of nodes of a media timeline may be utilized for efficient rendering of the media timeline. By improving rendering efficiency, the media timeline may be utilized on low resource devices, such as devices having limited hardware and/or software resources. For example, dynamic creation of the media timelines may include delayed creation of the nodes of the media timeline. The children of a parent node, for instance, need not be created until needed. The delayed creation of the nodes may be utilized to improve start-up and response times for media timelines having a significant number of nodes and/or a large amount of data for each node. For instance, a media player may be utilized to create and playback a playlist from a media

library that contains a significant number of selections. Creating such a playlist might require multiple queries to the media library, which may take a significant amount of time, processor and memory resources. By using delayed creation of the nodes, the playlist can be built on an "as needed" basis, thereby utilizing only as much processing and memory resources as required by the nodes needed at any one particular time. There are a wide variety of implementations that may be utilized for dynamic creation and/or loading of nodes of a media timeline.

[00123] FIG. 12 is an illustration of an example of a media timeline 1200 in an exemplary implementation in which the nodes of the media timeline are dynamically loaded based on metadata that is included in the nodes. The media timeline 1200 is illustrated as a tree structure that includes a root node 1202 and a plurality of nodes 1204-1216 that are children of the root node 1202. To render the media timeline 1200, the root node 1202 is first implemented and the metadata 1218 contained therein is examined. The metadata 1218 specifies a first grouping 1220 that includes nodes 1204, 1206. Therefore, when the root node 1202 is rendered, node 1204 and node 1206 are also loaded for rendering.

[00124] During or after the rendering of media referenced by the node 1206, metadata 1222 of node 1204 is examined that specifies a second grouping 1224 that includes node 1208 and 1210. Therefore, node 1208 and 1210 are loaded and media is output that is referenced by node 1210. Likewise, the metadata 1226 of node 1208 specifies a third grouping 1228 that includes nodes 1212, 1214, 1216. Therefore, nodes 1212, 1214, 1216 are loaded to output data referenced by nodes 1214, 1216 after the output of data referenced by node 1210 is completed.

[00125] FIG. 13 is an illustration of a media timeline 1300 in an exemplary implementation in which the nodes of the media timeline are defined and implemented on an "as needed" basis by a node source. In the previous implementation that was described in relation to FIG. 12, the media timeline 1200 was previously created and the nodes loaded on an "as needed" basis. In this implementation, the author defines a number of nodes that are children of a node 1302 of the media timeline 1300. The nodes are then created on an "as needed" basis during the rendering of the media timeline 1300. This is performed by attaching a node source 1304 module (node source) to the node 1302. The node source 1304 includes sufficient data such that, when executed, the node source 1304 may create the children of the node 1302, i.e. nodes 1306, 1308, and "fill-out" the properties of the nodes 1306, 1308, such as to supply metadata 1310, 1312 that defines properties and interrelationships as previously described in relation to FIG. 7. Therefore, when a particular one of the nodes 1306, 1308 is needed, the node source 1304 is implemented, e.g. called, to create the particular one of the nodes 1306, 1308. The node source 1304, for instance, may be executed to create the nodes 1306, 1308 in response to a request from the timeline source 124 of FIG. 1.

[00126] **Dynamic Changes to Nodes in a Media Timeline**

In one or more implementations, the media timelines are configured to be dynamically changed. For example, nodes of the media timeline may be removed, added or changed during the rendering of the media timeline by a timeline source. To provide for dynamic changes to the nodes, each node can generate events.

[00127] FIG. 14 is an illustration of a media timeline 1400 in an exemplary implementation in which events are provided by a node to such that changes that occur to

the media timeline 1400 may be communicated to nodes that may be affected by the changes. The media timeline 1400 includes a root node 1402 and a plurality of nodes 1404-1412 that are children of the root node 1402. Nodes 1408-1410 are utilized to reference media as previously described.

[00128] Each of the nodes 1402-1412 may generate events that may be utilized to inform other nodes of the media timeline 1400 that may be affected by changes to the node and/or changes to children of that node. For example, all events for node 1406 and any children of the node 1406, i.e. nodes 1410-1412, may be communicated to the root node 1402 and/or the author of the media timeline 1400. In other words, events in the media timeline 1400 may progress "up" the tree to the root of the tree. In this way, eventing may be utilized inform various nodes of the media timeline 1400 about dynamic changes to the timeline structure. Additionally, nodes of the media timeline 1400 may subscribe to events initiated by other nodes of the media timeline. Node 1408, for instance, may subscribe to receive events from node 1406 even though node 1408 is not a "parent" of the node 1406. Furthermore, components using the timeline, e.g. the media foundation 204 components of FIG. 2, can register to receive events initiated by any of the nodes. A variety of events 1414 may be supported by one or more nodes 1402-1412, examples of which are described as follows.

[00129] Node Added 1416

This event is issued when a node is added to the media timeline 1400. For example, node 1412 may be added to the media timeline 1400 to provide output of additional media referenced by the node 1412. Node 1406, when informed of the adding of node 1412, may issue the node added 1416 event such that it is communicated to the

root node 1402 through node 1404. Thus, in this example, each node 1402-1406 that is a parent of the newly added node 1412 is notified of events that are initiated by children of that node.

[00130] Node Removed 1418

The node removed 1418 event is issued when a node is removed from the media timeline 1400. Continuing with the previous example, node 1412 may be removed from the media timeline 1400 to remove the output of the media referenced by the node 1412. Node 1406, when informed of the removal of node 1412, may issue the node removed 1418 event such that it is communicated to the root node 1402 through node 1404. Thus, in this example, each node 1402-1406 that is a parent of the removed node 1412 is also notified.

[00131] Node Changing 1420

The node changing 1420 event is issued when metadata on a node of the media timeline 1400 is being changed. Node 1406, for instance, may include metadata, such as the metadata 704 described in relation to FIG. 7. Changes to the metadata may cause the node 1406 to issue the node changing 1420 event, which may be communicated to the application 202 of FIG. 2 and/or parents of the node 1406, i.e. nodes 1402, 1404. Thus, the node changing 1420 event may be utilized to inform other nodes and/or applications that utilize the node that changes are being made to the node 1406, and therefore respond according, such as to wait to render the node until a node changed 1422 event is received.

[00132] Node Changed 1422

The node changed 1422 event is issued when metadata on a node of the media timeline 1400 has been changed. Continuing with the previously example, node 1406

issued the node changing 1420 event such that other nodes and/or applications are informed that changes are being made to the node 1406. When the changes are complete, the node 1406 may issue the node changed 1422 event to inform the applications and/or nodes that the changes have been completed. In this way, the node 1406 may utilize the node changed 1422 event to inform that it is ready for rendering.

[00133] Remove Children 1424

The remove children 1424 event is issued when all of the children of a node are removed. Nodes 1410, 1412, for instance, may be removed from the media timeline 1400. Node 1406 issues the remove children 1424 event to inform the root node 1402 that the children, i.e. nodes 1410, 1412, of node 1406 have been removed. Thus, the remove children 1424 event may be utilized instead of issuing the node removed 1418 for each of the nodes 1410, 1412.

[00134] Node Source Added 1426, Node Source Removed 1428

The node source added 1426 event is issued when a node source is added to a node, such as the node source 1304 described in relation to FIG. 13. Likewise, the node source removed 1426 event is issued when a node source is removed from a node.

[00135] Node Sorted 1430

The node sorted 1430 event is issued when one or more nodes are sorted. For example, the media timeline 1400 may support a function in which the nodes 1402-1412 are sorted according to one or more criteria, such as chronologically, based on dependencies, and so forth. Therefore, the node sorted 1430 event may be initiated by the node 1406 when that node and/or children of the node 1406 (e.g., nodes 1410, 1412) are sorted.

[00136] Node Moved 1432

The node moved 1432 event is issued when a node is moved. For example, the node 1406 may be moved in the media timeline 1400 such that the node 1406 is a child of a different node, e.g. node 1402. Therefore, the node moved 1432 event may be initiated by the node 1406 and/or a parent of the node (e.g. the previous parent and/or the new parent node) when node 1406 is moved.

[00137] **Read-Only Media Timelines**

The author of a media timeline can mark all or a portion of the media timeline as read-only. This may be utilized to protect the functionality of the media timeline. In a first scenario, the author of the timeline does not want the user to change the media experience, such as to skip and/or delete advertisements. In another scenario, the author might want to dynamically change the media timeline, but does not want other components to modify it. In yet another scenario, the author might allow other components to set custom metadata on the timeline nodes, but not add new children to the timeline.

[00138] The media timeline can be customized to suit one or all of these read-only scenarios. Read-only media timelines may be implemented by creating a read-only wrapper of a media timeline. The read-only wrapper contains nodes which mirror the structure of the original timeline, i.e. are "cloned" from the nodes of the original timeline. The cloned nodes of the read-only media timeline may contain pointers back into the original timeline's nodes. Additionally, each of the cloned nodes may be configured to subscribe to events generated on the nodes of the original timeline. This allows the

cloned timeline's structure to be kept updated as the original media timeline changes, such as changes to the structure of the "tree" of the original media timeline.

[00139] The cloned nodes of the read-only media timeline may be configured to fail functions which allow the user to add/remove nodes to the read-only media timeline. When creating a read-only timeline, the author may also specify whether metadata for the cloned nodes should be modifiable. This design allows the author of the media timeline to modify the media timeline as much as desired while other components, e.g. applications that execute the read-only media timeline, have read-only or restricted access to the media timeline structure.

[00140] In an implementation, metadata 314 of the root node 302 of the media timeline 300 of FIG. 3 may be marked such that the media timeline 300 may not be edited by a user. In another implementation, a particular node and/or groupings of nodes of the media timeline may be marked as read-only. For example, referring again to FIG. 3, the metadata 320 of leaf node 308 may be marked as read-only. In another example, the metadata 318 of node 306 is marked as read-only such that node 306, leaf node 310 and leaf node 312 may not be edited.

[00141] **Exemplary Media Timeline Implementations**

The media timelines previously discussed may employ a variety of methods of storing and restoring timeline data, such as one or more Windows® Media Player Playlist files, eXecutable Temporal Language (XTL) files, and so on.

[00142] A media timeline, for instance, may be described as the following Windows® Media Player Playlist file identified by an ASX file extension.

<Asx Version = "3.0" >

```
<Entry>
<Ref href = "file://\\wmp\content\mpeg\Boom.mpe"/>
</Entry>
<Entry>
<Ref href = "\\wmp\content\Formats\MovieFile\chimp.mpg"/>
</Entry>
<Entry>
<Ref href = "file://\\wmp\content\mpeg\Boom.mpe"/>
</Entry>
</Asx>
```

This ASX file specifies three files for output, back to back. No start and stop times have been specified for the files. The ASX file may be represented by the media timeline 1500 shown in FIG. 15 that includes a sequence node 1502 and three leaf nodes 1504, 1506, 1508. Each of the leaf nodes 1504-1508 includes respective metadata 1510, 1512, 1514 that describes respective sources 1516, 1518, 1520 for media to be output by the media timeline 1500.

[00143] Another example of a media timeline is shown in the following XTL file.

```
<timeline>
<group type="video">
        <track>
<clip src="V1.wmv" start="0" stop="30" mstart="50" mstop="80" />
<clip src="V2.wmv" start="30" stop="40" mstart="0" />
        </track>
```

```
</group>
<group type="audio">
        <track>
<clip src="A1.asf" start="20" stop="40" mstart="0" />
<clip src="A2.asf" start="40" stop="60" mstart="0" />
</track>
</group>
</timeline>
```

This XTL file describes two tracks, e.g., streams, of media for output. One of the tracks is an audio track and the other is a video track.

[00144] The XTL file may be represented by the media timeline 1600 that is shown in FIG. 16 that includes a parallel node 1602 having two child sequence nodes 1604, 1606. In this example, sequence node 1604 has a major type 1608 filter set as "video" and sequence node 1606 has a major type 1610 filter set as "audio". Sequence node 1604 has two child leaf nodes 1612, 1614. Leaf node 1612 includes metadata that specifies a start time 1616 of "0", a stop time 1618 of "30", a media start 1620 of "50", and a media stop 1622 as "80". Leaf node 1614 include metadata that specifies a start time 1624 of "30", a stop time 1626 of "40", and media start 1628 as "0". It should be noted that leaf node 1614 does not include a media stop time, therefore the entire length of the media referenced by the leaf node 1614 will be output.

[00145] Sequence node 1606 also has two child leaf nodes 1630, 1632. Leaf node 1630 includes metadata that specifies a start time 1634 of "20", a stop time 1636 of "40", and a

media start 1638 of "0". Leaf node 1632 include metadata that specifies a start time 1640 of "40", a stop time 1642 of "60", and media start 1644 of "0".

[00146] FIG. 17 is an illustration of an exemplary implementation showing an output 1700 of first and second media over a specified time period that utilizes an effect to transition between the first and second media. In the illustrated example, A1.asf 1702 and A2.asf 1704 are two different audio files. A1.asf 1702 has an output length 20 seconds and A2.asf 1704 also has an output length 20 seconds. A cross fade 1706 effect is defined between the outputs of A1.asf 1702 and A2.asf 1704. In other words, the cross fade 1706 is defined to transition from the output of A1.asf 1702 to the output of A2.asf 1704. The cross fade 1706 effect is initiated at 10 seconds into the output of A1.asf 1702 and ends at the end of the output of A1.asf 1702. Therefore, the output of A2.asf 1704 is also initiated at 10 seconds. The cross fade 1706 is shown as inputting two different media, i.e. A1.asf 1702 and A2.asf 1704, and providing a single output having the desired effect.

[00147] FIG. 18 is an illustration of a media timeline 1800 in an exemplary implementation that is suitable to implement the cross fade 1706 effect of FIG. 17. The media timeline 1800 includes a parallel node 1802 having two children, i.e. leaf nodes 1804, 1806. The parallel node 1802 includes metadata that specifies a start time 1808 of zero seconds and a stop time 1810 of twenty seconds. The parallel node 1802 also includes a composite effect 1812 that describes a cross fade. The leaf node 1804 includes metadata indicating a start time 1814 of zero seconds and a stop time 1816 of twenty seconds. Leaf node 1806 includes metadata having a start time 1818 of ten seconds and a stop time 1820 of thirty seconds.

[00148] Leaf node 1804 also includes a pointer 1822 that references the A1.asf 1702 file described in relation of FIG. 17. Likewise, leaf node 1806 includes a pointer 1824 that references the A2.asf file 1704 that was described in relation to FIG. 17. Thus, when the media timeline 1800 is executed, the A1.asf 1702 file and the A2.asf file 1704 are output in a manner that employs the effect 1812 as shown in FIG. 17.

[00149] **Timeline Source**

FIG. 19 is a high level block diagram of a system 1900 in an exemplary implementation in which the system 1900, implemented in software, includes the media engine 208, media session 214 and timeline source 124 of the media foundation 204 to provide a presentation of media to the application 202. The timeline source 124 provides the media timeline 122 to act in a manner similar to the media source 210 as described in relation to FIG. 2. The timeline source 124, for instance, may be utilized to hide the intricacies of handling the media timeline 122 from the remaining components of the media foundation 204 of FIG. 2, such as the media session 214 and media processor 212.

[00150] The timeline source 124 manages a media presentation by interpreting the media timeline 122 to determine which of the media 206(g) of FIG. 2 is to be output at any one particular time. For example, the timeline source 124 may determine a topology for providing a presentation described by the media timeline 122. As previously described, the topology defines how data flows through various components for a given presentation. A topology may be a "full" topology that includes each of the components, e.g. software modules, used to manipulate the data such that the data flows with the correct format conversions between different components. A topology may also be a "partial" topology which describes the media and a source for the media, but does not

specify each component of the "full" topology, such as appropriate data conversion transforms, destinations, and the like.

[00151] The timeline source 124, for instance, may pass the topologies, full or partial, generated by the timeline source 124 from the media timeline 122 to the media processor 212. The media processor 212 is responsible for managing data flow in a topology. If the topology received from the timeline source 124 is a partial topology, a topology loader may be utilized to resolve the partial topology into a full topology as previously described. For example, media stream 228(1)-228(3) components (media stream) may be utilized to represent separate streams of media. Similar numbers are utilized to indicate that the media stream components 228(1)-228(3) correspond to the media stream 228 of FIG. 2. Additionally, transform objects such as codecs 1902(1), 1902(2), may be employed to provide appropriate data conversions between software components. Further, software components for supplying effects 1904(1)-1904(4) specified by the media timeline 122 may also be provided as was described in relation to FIGS. 8-11 may also be instantiated. Bitpumps 1906(1), 1906(2) are provided to communicate mediate output from the media processor 212 to respective media sinks 1908(1), 1908(2). The media sinks 1908(1), 1908(2) are utilized to represent "where" the presentation is to be presented (e.g. a window, disk file, and the like) and what happens to the presentation. Thus, through interpretation of the media timeline 12 by the timeline source 124, the timeline source 124 may provide one or more topologies that are suitable for providing a presentation of the media that is described by the media timeline 122, further discussion of which may be found in relation to FIG. 20.

[00152] **Timeline Source as a Media Source**

The timeline source is also a regular media source from which samples can be extracted. In other words, regular operations which can be performed by the media source 208 of FIG. 2 can also be performed on a timeline source. Example of media source operations include transport operations such as Start, Stop, Pause and other operations such as setting the rate, retrieving rate capabilities, and so on.

[00153] A timeline source can be created directly, such as by the application 202 of FIG. 2. The timeline source can also be created through the source resolver 216 as described in relation to FIG. 2. The source resolver 216 of FIG. 2, for instance, may utilize a URL, a byte stream, and so on, and creates the media source 210 and obtain media from it. A variety of media timeline formats may be resolved by a timeline source, such as ASX and XTL file formats that were described in relation to FIGS. 15-18. An example of pseudo code for an exemplary method of creating a timeline source may be represented as follows:

```
IMFSourceResolver* pSourceResolver = NULL;

hr = MFCreateSourceResolver( &pSourceResolver );
// error handling

hr = pSourceResolver->BeginCreateObjectFromURL( pwszURL, this, NULL );
// error handling
```

It should be noted that the pseudo code may be the same as an author may utilized to create other media source. Through use of the timeline source, data can be sourced for complex presentations with the same ease as a simple media source for applications which do not have intricate knowledge about media timelines.

[00154] In an implementation, the timeline source uses other internal media sources of the media foundation, such as media source 208 of the media foundation 204 of FIG. 2, to

retrieve media 206(g). In other words, the timeline source is not the originator of the media, but rather abstracts the provision of the media such that other software components need not be aware of where or how the media was obtained. Thus, the timeline source may hide the underlying media sources, such as by wrapping the underlying media source's presentation descriptors into its own presentation descriptors and wrapping underlying streams of media into its own streams.

[00155] The timeline source may also wrap events initiated by media sources it references. Status of a media source referenced by the timeline source may be communicated through events. The timeline source "listens" to the events from the media sources it resolves and uses internally and communicates relevant events to software components which interact with the timeline source, e.g. the media session 214 of FIG. 19. The timeline source may also provide events which are initiated by the timeline source itself.

[00156] In an implementation, multiple sources may be referenced to provide a presentation. In such an instance, operations that are performed on the timeline source are communicated to the media sources, if relevant. For example, consider a Start( ) call. When Start() is called on a media source, the calling software component expects to receive one source started event from the media source. Because the timeline source may be configured to appear as a media source, the timeline source adheres to the same protocols. For instance, a media timeline may include a parallel node and three child nodes that are to be played together. The timeline source first resolves the media sources for each of the three child nodes. The timeline source then retrieves a presentation descriptor from each of the nodes and creates a new presentation descriptor that wraps all the streams of media referenced by the underlying media sources. When a start call is

made on the timeline source with this presentation descriptor, the timeline source makes a start call on the three child nodes. In response to the three Start( ) calls, the timeline source receives three SourceStarted events, one from each of the child nodes. After receiving all three events, the timeline source initiates its own SourceStarted event. Thus, the software component that interacts with the timeline source receives only one event such that the three distinct sources of the media are hidden.

[00157] Timeline Source Service

Media sources may expose optional functionality through the use of services. The timeline source may also expose these services, as well as additional timeline source specific services through a timeline source interface, which will be referenced in the following discussion as "IMFTimelineSource". This interface is primarily for retrieving information about the underlying timeline. It also provides control for execution of the timeline. The timeline service interface may provide a variety of methods, examples of which are described as follows.

[00158] IMFTimelineSource::GetRootTimelineNode()

The GetRootTimelineNode method provides a way for an application to obtain the root node of the media timeline being rendered by the timeline source.

[00159] IMFTimelineSource:: GetNodes()

The GetNodes method returns the leaf nodes which are part of the given presentation.

[00160] IMFPresentationDescriptor

The timeline source initiates new presentations through the use of presentation descriptors. A presentation descriptor is represented by the IMFPresentationDescriptor

interface. In an implementation, this interface does not provide media timeline specific information. Rather, the presentation descriptor of a presentation may be accessed, during rendering, from the media processor and/or the media engine. In the case of a media timeline, however, an application may want to determine more than just which presentation is being rendered, but also which node of the media timeline is being rendered. A GetNodes function may be utilized to provide this capability. Thus, given a presentation descriptor, the GetNodes function may be utilized to locate nodes which correspond to the presentation.

[00161] IMFTimelineSource:: GePlaybackDuration()

This function may utilized to provide an indication of a playback duration of a particular node. As previously described, the output duration for a node need not be specified on the media timeline. This information, however, may be available during playback. Therefore, this function may be utilized to inform the application of the output duration that is calculated when the presentation is rendered.

[00162] IMFTimelineSource:: GetLocalTime()

The GetLocalTime method returns a time with respect to a given node. For example, when a media timeline is rendered, a presentation clock may be utilized to indicate a time for a particular presentation. This time, however, may not be sufficient for an application that interacts with a media timeline. For instance, the application may wish to determine progress of the rendering with respect to the entire media timeline. In another instance, the application may wish to determine a current duration of media that has been output by a particular node. Therefore, the GetLocalTime function may be

utilized to provide a translation from a presentation time to a desired time with respect to a given node.

[00163] IMFTimelineSource:: BeginNodePreroll () and IMFTimelineSource:: EndNodePreroll ()

In the timeline source, prerolling refers to an operation of resolving a media source for a node and reading the media source for rendering. As the timeline source interprets the presentations defined on the media timeline, the timeline source prerolls the nodes in the order for rendering. BeginNodePreroll, EndNodePreroll are asynchronous functions which allow the user, e.g. the application 202 of FIG. 2, of the timeline source to preroll a node on demand. In an implementation, this is independent from prerolling which is already provided by the timeline source automatically. For instance, if the application desires to "jump" to a node which is not yet ready for playback, the applications may first preroll the node. When the node is ready, the application may then jump to it and expect a quick response from the timeline source.

[00164] Execution plugin

The execution plugin allows the application to override the normal execution of the timeline source. For example the timeline source may render all the nodes in a sequence, one after the other. The application 202 of FIG. 19, however, may be configured as a media player that supports a shuffle mode, where nodes are randomly rendered. For such instances, the application may utilize the execution plugin to override the regular media timeline rendering by the timeline source. The execution plugin supports a variety of methods, examples of which are listed as follows:

GetFirst, which obtains a first leaf node for rendering;

GetLast, which is utilized to obtain a last leaf node for rendering;

GetNext, which is utilized to get a next node to be rendered; and

GetPrev, which is utilized to get a previous node for rendering.

For normal execution of the media timeline (e.g. rendering of the media timeline as described by the media timeline), the timeline source implements the above methods to interpret the media timeline. When an execution plugin is set on the timeline source, however, the methods may be delegated to the execution plugin such that execution plugin may be utilized to controls in what order the nodes are rendered.

[00165] **Interpreting a Media Timeline by a Timeline Source**

Given a media timeline, the timeline source may utilize a variety of techniques to divide the media timeline into distinct presentations. As previously described in relation to FIG. 2, a presentation refers to or describes the handling of media. A presentation can result in visually and/or audibly presenting media, such as a multimedia presentation in which both audio and accompanying video is presented to user via a window executing on a display device. A presentation can also result in writing media to a computer-readable medium such as a disk file. Thus, a presentation may include a wide variety of media scenarios, such as decoding, encoding and various effects, can take place as a result of the presentation and is not limited to scenarios in which multimedia content is rendered on a computer. Each presentation includes a presentation descriptor and a topology. A segment can be described as a interval of time where the components in a topology do not change, e.g. components included in the topology are not added and/or removed.

[00166] FIG. 20 is an illustration of an exemplary implementation in which a media timeline 2000 is shown that includes a sequence node 2002 and a plurality of leaf nodes 2004, 2006. The sequence node 2002 may be the same as or different from the sequence node 402 that was described in relation to FIG. 4. The plurality of leaf nodes 2004, 2006, is children of the sequence node 2002. Each of the plurality of leaf nodes 2004, 2006 includes respective metadata 2008, 2010 having respective pointers 2012, 2014 that reference respective media 2016, 2018 as previously described.

[00167] Additionally, the sequence node 2002 may include metadata 2020 that describes a transition effect 2022 to an output of the media 2016, 2018 referenced by the leaf nodes 2004, 2006. The transition effect 2022 is utilized to combine the media 2016, 2018 into a single output. Thus, the first input to the transition effect 1012 is supplied by leaf node 2004 and the next input to the transition effect 2022 is supplied by leaf node 2006.

[00168] FIG. 21 is an illustration of an exemplary implementation showing an output 2100 of media described by the media timeline 2000 of FIG. 20 over particular intervals of time and topologies generated by the timeline source to provide the described media output. The upper portion of FIG. 21 illustrates respective sources for media output during particular intervals of time. For instance, leaf node 2004 is illustrated as specifying output of media referenced by the leaf node 2004 between "0" and "20". Likewise, leaf node 2006 is illustrated as specifying output of media referenced by the leaf node 2006 between "10" and "30". As previously stated, the media 2016, 2018 referenced by the respective leaf nodes 2004, 2006 of FIG. 20 may include a variety of media, such as audio, video, multimedia, and so forth.

[00169] The transition effect 2022 is specified for output between the media referenced by leaf node 2004 and the media referenced by leaf node 2006. The transition effect 2022, when executed, combines media referenced by the leaf nodes 2004, 2006 to provide a single output of media having the applied effect.

[00170] The timeline source interprets the media timeline 2000 of FIG. 20 by examining the media timeline 2000 to determine individual presentations 2102(1)-2102(3) of the media timeline 2000 during which software components utilized to render the media do not change for a particular interval of time. As illustrated, presentation 2102(1) describes a time interval between "0" and "10" during which media referenced by leaf node 2004 is rendered. Therefore, during presentation 2102(1), only software components utilized to render the media 2016 are utilized. Likewise, presentation 2102(3) describes a time interval between "20" and "30" during which media referenced by leaf node 2006 is rendered. Therefore, for presentations 2102(1), 2102(3), software components utilized to render media referenced by respective leaf nodes 2004, 2006 do not change during a respective time interval.

[00171] Presentations may also describe rendering of a plurality of media. Presentation 2102(2), for instance, specifies media for rendering that is referenced by both leaf nodes 2004, 2006. Additionally, presentation 2102(2) also describes the transition effect 2022 to be applied to the media 2016-2018 of FIG. 20. Even though a plurality of media is described for rendering according to a transition effect 2022, software components utilized during the particular time interval, i.e. 10 to 20, corresponding to presentation 2102(2) do not change. In this way, the timeline source may divide the timeline 2000 of FIG. 20 into individual presentations 2102(1)-2102(3) such that each of the presentations

2102(1)-2102(3) describes rendering of media for a particular time interval during which software components utilized to provide the rendering do not change.

[00172] For each of the presentations 2102(1)-2102(3), the timeline source may then generate a respective topology 2104(1)-2104(3) for providing the described rendering. For example, topology 2104(1) references software components utilized to provide the rendering described by leaf node 2004. Topology 2104(2) references software components that are utilized to provide the rendering of leaf nodes 2004, 2006 and the transition effect 2022. Topology 2104(3) references software components that are utilized to provide the rendering described by leaf node 2006. The timeline source may then provide the topologies 2104(1)-2104(3) for rendering at the specified time interval, which is described in greater detail in the following implementation.

[00173] FIG. 22 is a simplified flow chart illustrating a procedure 2200 in an exemplary implementation in which a media timeline is interpreted by the timeline source for rendering of media as described by the media timeline. At block 2202, a media source examines a media timeline. The media timeline include a plurality of nodes, and at least two of the nodes reference respective media.

[00174] One technique which may be utilized to examine the media timeline utilizes a software module called the "node sorter". The node sorter walks the tree-structure of the media timeline and organizes the nodes and any corresponding effects into an array. For example, the nodes may be organized based on respective start and stop times that are specified in the metadata of each node, as was described in relation to FIG. 7. Every unique time value in the array identifies a particular interval of time. Nodes and/or

effects which have started before or at this time value are included in the particular interval of time.

[00175] At block 2204, based on the examining, the media timeline is divided into one or more presentations. Each of the presentations describes rendering of media for a particular interval of time described by the media timeline. As previously described, each of the presentations may reference a respective interval of time, during which, software components utilized to provide the described rendering do not change.

[00176] At block 2206, each software component referenced by a first one of the presentations is loaded. The first presentation, for instance, may be utilized to create a partial topology that references a source of desired media. The partial topology may be resolved to form a full topology that references each software component that, when executed, provide the media in the described manner. For example, the full topology may include transform objects, which were not specified in the partial topology, to provide data conversions. These topologies can then be given to the media processor to drive the data through the various components.

[00177] At block 2208, the first presentation is rendered. For example, each software component referenced in the first presentation may be executed as described by the presentation to process media referenced by the presentation. At block 2210, each software component referenced in a second one of the presentation is loaded. In an implementation, block 2210 is performed during the rendering of the first presentation of block 2208. At block 2212, when the rendering of the first presentation is completed, the second presentation is rendered. In this way, the timeline source may provide successive topologies that include software components that, when executed, provide the described

rendering for a particular interval of time. Thus, through utilization of the timeline source, each software component referenced by the media timeline need not be loaded and/or created during loading of the media timeline, but rather may be created and/or loaded on an "as needed" basis.

[00178] **Timeline Source Interaction in the Media Foundation**

The timeline source may interact with other media foundation components, such as the media foundation components that were described in relation to FIG. 2, to render the media timeline. In the following implementations that are described in relations to FIGS. 23 and 24, generation and rendering of the media timeline is described that also references the software components of the media foundation that were described in relation to FIGS. 2 and 19.

[00179] FIG. 23 is a flow chart illustrating a procedure 2300 in an exemplary implementation in which generation of a media timeline and interpretation of the media timeline by a timeline source is described. At block 2302, the application 202 calls the media engine 208 and passes a URL that references a media timeline file. The media timeline file may have a variety of formats, such as ASX and XTL formats that were described in relation to FIGS. 15-18.

[00180] At block 2304, the media engine 208 invokes the source resolver 216 to resolve the URL to a media source. At block 2306, the source resolver 216 resolves the URL to a bytestream. At block 2308, the source resolver 216 invokes the bytestream plugin 236 that is registered for the specified format of the bytestream. At block 2310, the bytestream plugin 236 creates the media timeline 122 by parsing the bytestream.

[00181] At block 2312, the bytestream plugin 236 creates the timeline source 124 and specifies the media timeline 122 as an input. As previously discussed, the timeline source 124 may act as a media source to provide media for rendering.

[00182] At block 2314, the timeline source 124 interprets the media timeline 122 to identify a first presentation. For example, the timeline source 124 may examine the media timeline 122 and divide it into one or more presentations. At block 2316, the timeline source 124 resolves media sources for the first presentation using the source resolver 216 and creates a presentation descriptor and a topology for the first presentation. The topology may include effects that are specified by the media timeline 122. The presentation descriptor may be utilized to distinguish each presentation, one from another. At block 2318, the bytestream plugin 236 returns control of the timeline source 124 to the source resolver 216, and at block 2320, the source resolver 216 returns control of the timeline source 124 to the media engine 208. Thus, the media engine 208 is ready to initiate rendering of the media timeline 122 by the timeline source 124, which is described in greater detail in the following implementation.

[00183] FIG. 24 is a flow chart illustrating a procedure 2400 in an exemplary implementation showing rendering of the media timeline 122 of FIG. 22 by the timeline source 124. At block 2402, the media engine retrieves a first presentation descriptor and a corresponding partial topology from the timeline source 124. As previously stated, the presentation descriptor is utilized to identify a particular one of the presentations identified at block 2314 of FIG. 23.

[00184] At block 2404, the media engine begins to resolve the partial topology by obtaining a destination for media referenced by the first presentation from the application

202. For example, the partial topology of block 2402 may specify a source of the media, but may not supply a destination for the media. Therefore, the media engine 208 may obtain the destination 224 from the application 202. The destination 224 may be represented by the media engine 208 through one or more media sinks 220, 222. At block 2406, the media engine 208 completes resolution of the partial topology to a full topology. The media engine 208, for instance, may supply one or more transform objects that provide data conversion, supply one or more effect objects to provide effects specified in the partial topology, and so forth.

[00185] At block 2408, the media engine 208 sets the full topology on the media session 214. At block 2410, the media session 214 sets the full topology on the media processor 212. As previously stated, the media processor 212 may be used to drive a given presentation, and the media session 214 utilized to schedule multiple presentations. The media session 214, for instance, may change topologies that are rendered by the media processor 212.

[00186] At block 2412, the media processor 212 is configured to "drive" the topology and calls "start" on the timeline source 124. At block 2414, the timeline source 124 initiates all media sources referenced in the current presentation, e.g. the first presentation. If multiple media sources are present, the timeline source 124 aggregates events from the multiple media sources to provide a single corresponding event to the media processor 212.

[00187] At block 2416, the timeline source 124 determines a next presentation specified by the media timeline 122. For example, the timeline source 124 may obtain another presentation that was interpreted at block 2314 of FIG. 23, interpret another presentation

from the media timeline 122 during the execution of the media sources at block 2414, and so forth. The timeline source 124, for instance, may create a presentation descriptor and a topology for the next presentation.

[00188] At block 2418, the timeline source 124 initiates a "new presentation" event to the media session 214 through the media processor 212. At block 2420, the media session 214 queues the topology for the next presentation. At block 2422, when the rendering of the current presentation is complete, e.g. media referenced by the first presentation has been processed by the software components as described, the next topology is set on the media processor 212 by the media session 214. At block 2424, the timeline source 124 then receives another start call for the next presentation. Thus, the timeline source 124 may initiate the media sources as was previously described in relation to block 2414. The procedure 2400 is completed when each presentation is rendered.

[00189] **Exemplary Operating Environment**

The various components and functionality described herein are implemented with a number of individual computers. FIG. 25 shows components of a typical example of a computer environment 2500, including a computer, referred by to reference numeral 2502. The computer 2502 may be the same as or different from computer 102 of FIG. 1. The components shown in FIG. 25 are only examples, and are not intended to suggest any limitation as to the scope of the functionality of the invention; the invention is not necessarily dependent on the features shown in FIG. 25.

[00190] Generally, various different general purpose or special purpose computing system configurations can be used. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not

limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, network-ready devices, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[00191] The functionality of the computers is embodied in many cases by computer-executable instructions, such as software components, that are executed by the computers. Generally, software components include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Tasks might also be performed by remote processing devices that are linked through a communications network. In a distributed computing environment, software components may be located in both local and remote computer storage media.

[00192] The instructions and/or software components are stored at different times in the various computer-readable media that are either part of the computer or that can be read by the computer. Programs are typically distributed, for example, on floppy disks, CD-ROMs, DVD, or some form of communication media such as a modulated signal. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory.

[00193] For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different

storage components of the computer, and are executed by the data processor(s) of the computer.

[00194] With reference to FIG. 25, the components of computer 2502 may include, but are not limited to, a processing unit 2504, a system memory 2506, and a system bus 2508 that couples various system components including the system memory to the processing unit 2504. The system bus 2508 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISAA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as the Mezzanine bus.

[00195] Computer 2502 typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by computer 2502 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. "Computer storage media" includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by

computer 2502. Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more if its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[00196] The system memory 2506 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 2510 and random access memory (RAM) 2512. A basic input/output system 2514 (BIOS), containing the basic routines that help to transfer information between elements within computer 2502, such as during start-up, is typically stored in ROM 2510. RAM 2512 typically contains data and/or software components that are immediately accessible to and/or presently being operated on by processing unit 2504. By way of example, and not limitation, FIG. 25 illustrates operating system 2516, application programs 2518, software components 2520, and program data 2522.

[00197] The computer 2502 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 25 illustrates a hard disk drive 2524 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 2526 that reads from or writes to a removable, nonvolatile magnetic disk 2528, and an optical disk drive 2530 that reads from or writes to a

removable, nonvolatile optical disk 2532 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 2524 is typically connected to the system bus 2508 through a non-removable memory interface such as data media interface 2534, and magnetic disk drive 2526 and optical disk drive 2530 are typically connected to the system bus 2508 by a removable memory interface.

[00198] The drives and their associated computer storage media discussed above and illustrated in FIG. 25 provide storage of computer-readable instructions, data structures, software components, and other data for computer 2502. In FIG. 25, for example, hard disk drive 2524 is illustrated as storing operating system 2516', application programs 2518', software components 2520', and program data 2522'. Note that these components can either be the same as or different from operating system 2516, application programs 2518, software components 2520, and program data 2522. Operating system 2516', application programs 2518', software components 2520', and program data 2522'are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 2502 through input devices such as a keyboard 2536, and pointing device (not shown), commonly referred to as a mouse, trackball, or touch pad. Other input devices may include source peripheral devices (such as a microphone 2538 or camera 2540 which provide streaming data), joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 2502 through an input/output (I/O) interface 2542

that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB). A monitor 2544 or other type of display device is also connected to the system bus 2508 via an interface, such as a video adapter 2546. In addition to the monitor 2544, computers may also include other peripheral rendering devices (e.g., speakers) and one or more printers which may be connected through the I/O interface 2542.

[00199] The computer may operate in a networked environment using logical connections to one or more remote computers, such as a remote device 2550. The remote device 2550 may be a personal computer, a network-ready device, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 2502. The logical connections depicted in FIG. 25 include a local area network (LAN) 2552 and a wide area network (WAN) 2554. Although the WAN 2554 shown in FIG. 25 is the Internet, the WAN 2554 may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the like.

[00200] When used in a LAN networking environment, the computer 2502 is connected to the LAN 2552 through a network interface or adapter 2556. When used in a WAN networking environment, the computer 2502 typically includes a modem 2558 or other means for establishing communications over the Internet 2554. The modem 2558, which may be internal or external, may be connected to the system bus 2508 via the I/O interface 2542, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 2502, or portions thereof, may be stored in the remote device 2550. By way of example, and not limitation, FIG. 25 illustrates remote

software components 2560 as residing on remote device 2550. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[00201] **Conclusion**

Although the invention has been described in language specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the claimed invention.